White Paper:
Graphics Processing Units in Enterprise Architectures

April 20, 2009

Prepared by:
Celestech, Inc.
4505 E. Chandler Blvd.
Suite 155
Phoenix, AZ 85048
(480) 940-1640

Table of Contents

Abstract

Commodity Graphics Processing Units (GPUs) are enabling a revolution in computing by providing ultra high speed, teraflop processing power with a single, standard personal computer board. GPUs can be combined to form small processing arrays with staggering computational power at a relatively low price. This exceptional processing power at such a low cost will enable moving new algorithmic approaches that were previously too computationally intensive to consider outside of the Research and Development (R&D) lab into routine operational analysis.

In the workstation environment, performance increases of at least one or two orders of magnitude are common on GPU-enabled applications.

When GPUs are integrated at the enterprise level, they can provide aggregated performance increases that can translate to users having access to cost effective supercomputer capability. Abstracting the GPU resources as a service to other applications hides the complexity associated with programming them. This abstraction also allows service requests to be divided up among multiple GPU processors (much more than could be supported in a workstation environment) in a manner that is transparent to the user and the application-layer, enabling enterprise integration without making major changes to the service-oriented or enterprise architecture.

GPU programming techniques have a steep learning curve, but with experienced professional assistance, the recasting of computationally intensive legacy applications for the GPU can be significantly accelerated and implemented at a relatively low cost. Celestech engineers have mastered these programming techniques and possess the experience necessary to accelerate development of GPU-enabled applications and the molding of legacy code into highly efficient parallel implementations.

Introduction

Enterprise computing environments in certain niches are making use of the GPU's computational power. Supercomputer cluster researchers (such has Tokyo Technical University's Tsubame Super Computer) and the financial sector are early adopters of this technology. Adapting this technology for enterprise architectures is lagging behind workstation implementations. This white paper will examine the reasons for this lag as well as offer some alternatives for maximizing the GPU's utility in enterprise data centers.

GPUs by the Numbers

Before launching into a discussion about how to incorporate GPUs into enterprise applications, it makes sense to understand the GPU and compare and contrast it with traditional CPU implementations.

The chart below illustrates various performance attributes of today's common CPUs and a typical GPU. As you can see from the numbers, the raw performance of the GPU stretches the limits of comprehension. A trillion is a very big number.

Processor	**Cores**	**Cache**	**Clock Speed**	**GFLOPS**	**Memory Bandwidth**
Intel Core tm i7-965	4	8MB	3.2 GHz	51.2	64 GB/sec
Intel Itanium© 9152M	2	24MB	1.6 GHz	13.3	34 GB/sec
Intel Core 2 Quad QX9770	4	12MB	3.2 GHz	51.2	26 GB/sec
NVIDIA GeForce GTX 285	240	256KB	648 MHz	**1,060.0**	159 GB/sec

The raw power of the GPU far exceeds traditional CPU architectures in floating-point applications and is creating a high interest among users with computationally intensive applications. As shown in the table, the GPU wins the GigaFLOP drag race by a country mile. The paltry amount of cache that the GPU possesses is compensated for in that the application developer has complete control over how the cache is utilized.

Pricing is another positive aspect of the GPU. The current price for the GPU card shown in the table is under $400. By comparison the i7 processor chip retails for about $1000. If you factor in the relative cost and floating-point performance delta, the GPU is more than 50-times less expensive than traditional CPU platforms. GPUs also provide additional savings in terms of lower power consumption and heat generation as well as smaller form factors.

Being a dedicated math coprocessor, the GPUs will never replace general purpose CPUs, but they can provide significant value where substantial computational performance is required.

GPUs have extremely high compute density. In certain applications, Field Programmable Gate Arrays (FPGAs) may achieve a higher per chip computational density, but FPGAs have significantly higher cost and power consumption and a higher degree of programming difficulty.

What GPUs Do Well

The GPU is a highly specialized, purpose-built, floating-point processor of remarkable speed, initially designed to process real-time graphics. The core architecture is a single instruction multiple data (SIMD) parallel computer. Applications that perform the same operations on multiple data elements, particularly if these are floating-point operations, are ideal candidates for GPU acceleration. Performance improvements of several orders of magnitude are achievable with computationally intensive floating point applications.

GPUs are not a panacea for speeding up every application. Input/Output (I/O) bound applications such as collecting and forwarding sparse data elements from disparate databases are not good candidates for GPU acceleration and can actually run slower on a GPU.

The Good News and the Bad News

The good news is that the advertized performance the GPU can be realized in numerous applications. Celestech has developed applications for nonlinear hyperspectral image processing that approach the theoretical performance of the host GPU platforms. Other application domains have achieved similar results.

The bad news is that all of this power is not yet mindlessly integrated into the underlying fabric of computational environments. Because the GPU is a new, very different technology, no automated means of optimizing applications yet exists. Presently in order to realize the GPU's potential, applications have to be designed with the GPU in mind. To extract the most from the GPU, programmers have to think in a parallel paradigm. Machines are currently unable to redefine algorithms automatically on any sort of abstract, logical level, and current compiler optimization techniques are far from achieving this capability. Until this problem is solved, GPU enabled applications will require a GPU application developer to fully exploit the power of the GPU.

Some vendors are offering generic plug-ins for applications like Matlab and LabVIEW. These plug-ins allow a user to get some power out of the GPU without having any understanding of how the GPU works and they do represent the first steps down a path that may ultimately lead to greater GPU integration. Unfortunately the performance increase achieved by these plug-in applications is typically well short of the performance capability inherent in the GPU. It's somewhat like putting 4 compact spare tires on a Ferrari. It has the potential for tremendous performance, but you are never going to realize that potential.

Legacy Applications and the GPU

Acceleration of legacy applications is the principal area of interest for a potential enterprise GPU implementation. Integrating GPUs in a legacy environment will produce a high rate of return on a relatively modest investment. Because the GPU's forte is floating-point calculations, incorporating a GPU into a legacy application need not mean a complete rewrite. Only those portions of the application that are computationally bound need to be ported to the GPU. The remainder of the application can remain untouched. In a typical application this section of the code is usually a relatively small percentage of the total. The rest of the code is generally devoted to operator interactions, data collection and storage and other control type functions. All of those portions of the code can remain intact with just the computational core of the algorithm being transported to the GPU environment. This approach allows applications to take advantage of both types of processors (CPUs and GPUs) having each assigned the tasks that they are best suited to perform.

The integration of GPU hardware and tailored applications can be implemented in a way that is transparent to the user. In a workstation environment this means integrating a GPU card in every computer workstation. Because GPUs are an inexpensive commodity item, this approach would not be cost prohibitive. The parallelism inherent in GPU implementations can typically be spread across multiple GPU cards with a linear increase in performance. In other words, two GPU cards will run the application twice as fast. So having access to multiple GPU cards is desirable.

Enterprise Computing and the GPU

Enterprise computing offers interesting possibilities for integrating GPUs and allowing users to transparently take advantage of their power. At the enterprise level servers are typically consolidated in data centers, data cubes, or clusters accessed by a large number of users via remote connections. Within the data center large amounts of storage, network bandwidth, and processing power are concentrated to provide data services to the user community.

Requiring user cognizance of the hardware environment in the enterprise services/server environment can potentially be very problematic. Any GPU integration would need to be transparent to the applications and users. Using a Service-Oriented Architecture (SOA), hardware transparency can be implemented by abstracting the GPU-enhanced applications as services.

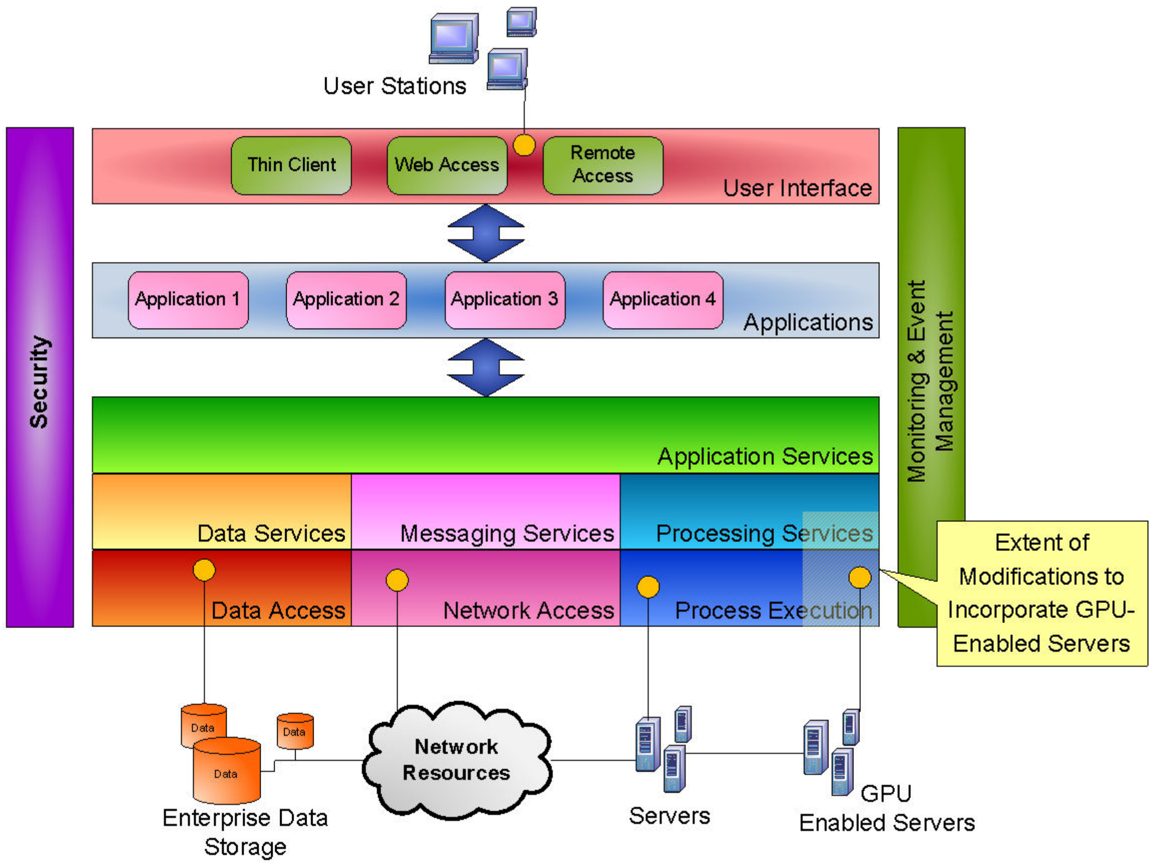

Figure 1 - Notional SOA incorporating GPU-Enabled Servers

At the hardware level GPU-enabled servers resemble their workstation cousins. Each GPU-enabled server will consist of a CPU (server) interconnected to a collection of GPU boards via high-speed interface. The CPU provides control of the GPU coprocessor resources and manages the I/O to the rest of the data center. The CPU also facilitates communication with outside applications and can provide the service interface to the rest of the SOA.

By encapsulating the GPU's accelerated application functions within a service defined within the SOA framework, the complexity is isolated from the users and the other applications. Figure 1 shows a notional SOA that includes GPU-accelerated functions. The GPU-enhanced services allow the application layer, and ultimately the users, to leverage the GPU's processing capability without having to be directly cognizant of how it works.

The GPU-enhanced services can also leverage multiple GPU hardware platforms to perform different segments of the same user service request. The GPU services can pool all the GPU resources, divide a service request across those resources (much like adding a second GPU card to workstation) and allow multiple GPU hardware resources to complete different sections of the same service request at the same time. This method is similar to a distributed query to a database. For service request that involve large amounts of data and large quantities of computation, this approach can provide an exceptional increase in performance.

Consider the following example. Let's assume that we have a SOA that serves a large user community. The servers for the services are hosted in large data center that consists of multiple racks of traditional servers, GPU enabled servers, and large storage arrays. The SOA supports a mix of data intensive and computation intensive applications for users running on thin clients. Assume that one of the applications supported is a Geographic Information Service (GIS) that allows users to go to a geographic location and display layered information about that location. Let us also assume that the data for each of the layers is a set of multi-gigabyte files in different formats with some of them requiring transformation into a compatible format in order to be displayed.

If GPU enhanced transformation algorithms are implemented within the SOA, the GIS application will make a request to a data service for the appropriate layers of data. The data service will identify the appropriate data stores within the data center and invoke the transformation service to convert the raw data into the appropriate format for display. The transformation service software will divide the data request across a set of GPU-enabled servers that process the segmented elements of the data set in parallel. The operation completes rapidly on each GPU server and the transformation service returns the information to the requesting application. From the user's perspective the data transformation task for an extremely large quantity of data will be performed nearly instantly. The abstraction of the SOA enables the complexities of the GPU to be hidden from all but a small portion of the code and also allows all users to transparently share a pool of GPU resources and have multiple teraflops of computational power at their disposal.

The addition of GPU-enhanced services to an existing SOA can be transparent to the users – a new GPU-enhanced service can simply replace an existing service within the

SOA. The only outward indication to the other applications and the user is that the service is now significantly faster than it used to be.

Under the Hood

Most Enterprise application architectures incorporate a distributed processing model where users have a local client or log into a server and then make use of the enterprise resources from that access point. In typical SOA or other enterprise architecture models, the enterprise resources are abstracted from the application software in such a way that the underlying infrastructure is nearly transparent to the applications layer products. These are all beneficial features for GPU integration into the enterprise as they enable the GPU applications/services to replace existing, compute-bound sections of applications without disrupting large portions of the enterprise environment.

At the lower layers, GPU integration into an enterprise architecture could look something like the notional view of Figure 2.

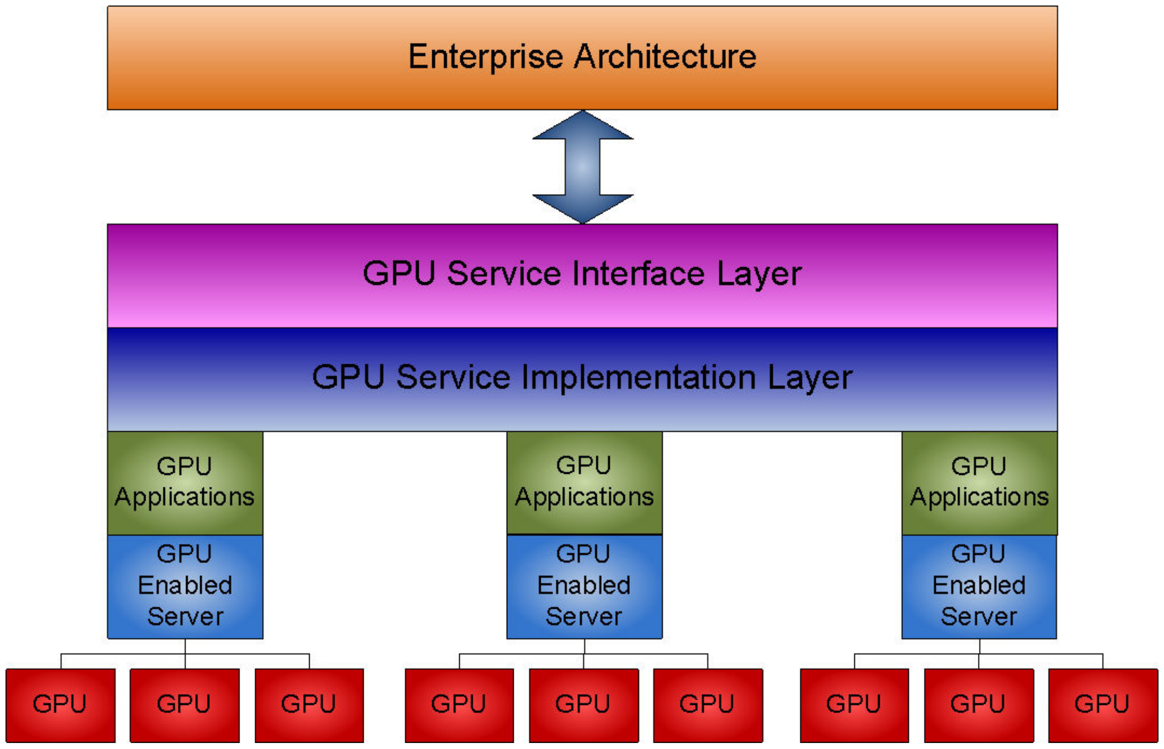

Figure 2 – Notional GPU Implementation, Management and Service Layers

At the hardware layer GPU cards or appliances are plugged into servers that are connected to the network fabric, providing the hardware connectivity between the GPU-enabled servers, the users, and the data. Data is stored in storage arrays and is available to network resources.

At the lowest GPU application layer, the GPU accelerated algorithms reside on the GPU servers. These are purpose-built collections of the computationally intensive software routines that the enterprise requires. This software is optimized for the GPU platform. These core GPU-enhanced application functions can be invoked by other elements of the enterprise architecture as well. The computational speed of the GPU routines will alleviate the computational bottlenecks that exist in non-GPU enable enterprise architectures.

The GPU service implementation layer allocates the service request across available GPU resources. The collection of GPU servers can be managed either hierarchically or in a shared, distributed fashion. This layer of software will process the service requests received from the enterprise architecture service interface. These requests might be something like “perform this geodetic transform on the following sets of data”. After a request is received, the GPU service implementation layer evaluates the processing requested and the scope of the data involved in each request. It assesses the resource needs and allocates an optimal number of GPUs and GPU servers to complete the task. Each GPU server will be assigned to process a subset of the total data requested. Because of the SIMD nature of GPUs, this sort of distributive ability is a by-product of the GPU application optimization process. As these requests are completed the data is returned to the application layer via the appropriate service interface. This distribution of requests has the effect of putting multiple teraflops of computational power in the hands of every enterprise user.

In order to make these GPU functions available to the enterprise architecture they need to present a service interface to the rest of the enterprise. This is handled at the service interface layer. The GPU service interface layer is ideally plug-compatible with the legacy service interface that the GPU-enabled functions are replacing. By making this interface identical to the legacy interface, the remainder of the enterprise architecture is oblivious to the GPU-enabled implementation and can remain unchanged.

Getting Started

Getting started with implementing a GPU based solution can be easy and relatively inexpensive. A GPU-enabled workstation can be purchased for a couple of thousand dollars. A group of 4 GPU-enabled computers with 16 teraflops of computational power can be purchased for approximately $10K. An enterprise-oriented, rack-mounted GPU device with 4 teraflops of processing power will cost approximately $8,000 and require a server to connect to it.

The first step in implementing enterprise GPU-enabled services is to identify an initial candidate service for implementation of a prototype. The candidate service/function should be computationally intensive and frequently used. Computational bottlenecks are typically well-known within the organizations that maintain the SOA, so candidate identification is usually straightforward.

Once a likely candidate for implementation is identified, the next step is to prototype the software on a GPU-enabled workstation and benchmark the performance against a CPU-based implementation. This will provide empirical evidence of the GPU algorithm's performance against the legacy CPU implementation.

Assuming that the GPU's performance increase justifies moving forward, the next step is to integrate the GPU algorithm into the existing service. The service should be thoroughly tested using production data and rolled out into production in accordance with the organization's deployment methodology.

The key to success is having knowledgeable engineering assistance during the prototype implementation phase. This step is the most problematic and is the point where GPU implementation activities encounter the greatest amount of difficulty. The software development skills required to create an optimal GPU implementation are not very common among the software development community. The learning curve for these devices is very steep and it is not uncommon for programmers to take a year or more to achieve any degree of proficiency in GPU algorithm design and optimization. Since most organizations lack the patience for this learning curve, the GPU projects are often canceled long before they achieve success. There is a simple solution - hire experienced help.

Celestech is one of the few companies in the community with significant experience and demonstrated success with GPU algorithm development and programming. Our programmer's experience and proven skills can significantly accelerate implementation timelines, reduce cost and schedule, and mitigate risk associated with GPU software development. Our software engineers have been working with the GPU platform for over two years and have developed the requisite abstract thinking skills required to view the problem from a parallel perspective. Celestech's team has developed performance-leading algorithm implementations for some of computer science's classic problems including the K-Nearest Neighbor (KNN) search and Dijkstra's shortest path algorithm.

In working with the GPU, we have learned that the algorithm and performance optimization process can be counter-intuitive. We have also learned that very small changes can have a very large impact on performance and that a detailed understanding of how the software and data are interacting with the hardware is required to realize the full performance potential of the GPU devices.

For additional information on our capabilities in this area, please contact Rusty Topping at (480) 940-1640 ext. 201 or via email at Rusty.Topping@celestech.com